WE CLAIM:

1. A transaction queue for an agent that operates according to a dynamic priority scheme, the transaction queue operating according to a default priority scheme and engaging a second priority scheme when a congestion event is detected.

2. In an agent, a management method for external transactions, comprising:

queuing data of a plurality of read requests,

for each queued request, storing data of a blind prefetch transaction associated with the respective request,

when a transaction congestion event occurs, disabling selected stored prefetch requests.

3. The management method of claim 1, wherein the transaction congestion event occurs when a number of queued requests exceeds a predetermined threshold.

4. The management method of claim 1, wherein the transaction congestion event occurs when a queue that stores the queued request becomes full.

5. The management method of claim 1, wherein the transaction congestion event occurs when a measured latency of posted transactions exceeds a predetermined threshold.

6. In an agent, a management method for external transactions, comprising:

queuing data of a plurality of external bus transactions,

for at least one queued transaction, storing data of a blind prefetch transaction in association with the respective transaction,

when a transaction congestion event occurs, disabling the blind prefetch transaction.

7. The management method of claim 6, wherein the transaction congestion event occurs when a number of queued requests exceeds a predetermined threshold.

8. The management method of claim 6, wherein the transaction congestion event occurs when a queue that stores the queued request becomes full.

9. The management method of claim 6, wherein the transaction congestion event occurs when a measured latency of posted transactions exceeds a predetermined threshold.

10. In an agent, a management method for external transactions, comprising:

queuing data of a plurality of read requests, certain read requests related to executions being performed by an agent core, certain other read requests related to data being prefetched,

when a transaction congestion event occurs, disabling the prefetch requests.

11. The management method of claim 10, wherein the transaction congestion event occurs when a number of queued requests exceeds a predetermined threshold.

12. The management method of claim 10, wherein the transaction congestion event occurs when a queue that stores the queued request becomes full.

13. The management method of claim 10, wherein the transaction congestion event occurs when a measured latency of posted transactions exceeds a predetermined threshold.

14. In an agent, a multi-mode management method for external transactions, comprising:

queuing data of a plurality of core read requests

for each core read request, storing data of a blind prefetch transaction associated with the respective core read request,

queuing data of prefetch requests related to patterns of core read requests,

in a first mode, when a transaction congestion event occurs, disabling the blind prefetch transactions,

in a second mode, when a transaction congestion event occurs, disabling the prefetch requests.

15. The management method of claim 14, wherein the transaction congestion event occurs when a number of queued requests exceeds a predetermined threshold.

16. The management method of claim 14, wherein the transaction congestion event occurs when a queue that stores the queued request becomes full.

17. The management method of claim 14, wherein the transaction congestion event occurs when a measured latency of posted transactions exceeds a predetermined threshold.

18. A transaction queue, comprising:

a controller,

a plurality of queue registers, each having an address field and status fields associated with a pair of transactions related to the address, wherein, in response to a congestion event, the controller modifies one of the status fields in a register to invalidate the respective transaction.

19. The transaction queue of claim 18, wherein

the transaction queue stores core read requests, blind prefetch requests and patterned prefetch requests, and

the invalidated transaction is a blind prefetch request.

20. The transaction queue of claim 19, wherein, when there are no valid blind prefetch requests, the controller invalidates a patterned prefetch request.

21. A bus sequencing unit of an agent, comprising:

an arbiter,

an internal cache coupled to the arbiter,

a transaction queue coupled to the arbiter and storing data of external transactions to be performed by the agent, the transactions selected from the group of core read requests, blind prefetch requests and patterned prefetch requests, and

an external bus controller coupled to the transaction queue,

wherein, in response to a congestion event in the bus sequencing unit, the transaction queue invalidates selected transactions.

22. The bus sequencing unit of claim 21, wherein the selected transaction is a blind prefetch request.

23. The bus sequencing unit of claim 21, wherein, when there are no blind prefetch requests in the transaction queue, the selected transaction is a patterned prefetch request.

24. A congestion management method for external transactions, the external transactions stored in the transaction queue in pairs and taken from the set of a core read requests-blind prefetch request pair and a patterned prefetch request pair, comprising:

receiving a new request at a transaction queue,

determining whether the transaction queue has space to store the new request,

if not,

removing a pair of patterned prefetch request from the transaction queue and storing the new request in the transaction queue.

25. The congestion management method of claim 24, further comprising invalidating a blind prefetch request.

26. The congestion management method of claim 23, further comprising invalidating a first patterned prefetch request when a second patterned prefetch request in the pair has been posted.